Summary of Last Lecture

- Normalization
  - 2NF
    - a. should be in 1NF
    - b. No partial dependency
  - 3NF
    - a. should be in 2NF
    - b. No transitive dependency

Boyce Code Normal Form (BCNF)

BCNF is an extension of Third Normal Form on strict terms.

BCNF states that :

- The relation should be in 3NF
- For any functional dependency, $X \rightarrow A$, X must be a super-key.

Boyce Code Normal Form (BCNF)

R = {emp_id, emp_dept, nationality, dept_type, dept_no }

Key = {emp_id}

F = { emp_id → emp_dept,
emp_id → nationality,
emp_dept →dept_type,
emp_dept → dept_no }

Is the relation in BCNF?

R1 = {emp_id, emp_dept,nationality}
R2 = {emp_dept, dept_type, dept_no}

Boyce Code Normal Form (BCNF)

Example

R = { author, nationality, book_title, category, no_of_pages}

Key = {author}

F = {author →nationality,

author → book_title,

book_title → category,

book_title → no_of_pages }

Is the above relation in BCNF?

R1={author, nationality, book_title}

R2={book_title, category, no_of_pages}

Decomposition Algorithm

These are two important properties associated with decomposition.

- Lossless Join
- Dependency Preservation

Decomposition Algorithm

Lossless Join -

A decomposition of a relation R into schemes Ri ($1 \leq i \leq n$) is said to be a lossless join decomposition or simply lossless if for every relation (R) the natural join of the projections of R gives the original relation R; i.e.,

$$R = R1 \bowtie R2 \bowtie \ldots \bowtie Rn$$

If $R \subseteq R1 \bowtie R2 \bowtie \ldots \bowtie Rn$ then the decomposition is called lossy.

Decomposition Algorithm

Example: R

Model Name	Price	Category
a11	100	Canon
s20	200	Nikon
a70	150	Canon

Model Name	Category
a11	Canon
s20	Nikon
a70	Canon

Price	Category
100	Canon
200	Nikon
150	Canon

Decomposition Algorithm

R1 ⋈ R2

Model Name	Price	Category
a11	100	Canon
a11	150	Canon
s20	200	Nikon
a70	100	Canon
a70	150	Canon

R

Model Name	Price	Category
a11	100	Canon
s20	200	Nikon
a70	150	Canon

Decomposition Algorithm

Dependency Preserving

Given a relation R where F is a set of functional dependencies,

R is decompose into the relations

R1,R2,...,Rn

with the functional dependencies

F1,F2,...,Fn

Then this decomposition of R is dependency preserving if the

closure of F1∪ F2 ∪ ... ∪ Fn is identical to F+

Decomposition Theorem

A decomposition of relation R<(x,y,z),F> into R1<(x,y),F1> and R2 <(x,z),F2> is:

a) dependency preserving if every functional dependency in R can be logically derives from the functional dependencies of R1 and R2 i.e. $(F1 \cup F2)^+ = F^+$

b) is lossless if the common attribute x of R1 and R2 form a key of at least one of these i.e. $x \rightarrow y$ or $x \rightarrow z$

Decomposition Theorem

Example -

Let R(a,b,c) and F = {$a \rightarrow b$ }

Check weather above relation is lossless and dependency preserving if decompose as

a. R1 (a,b) and R2 (a,c) :

lossless, dependency preserving

b. R1 (a,b) and R2 (b,c) :

lossy, dependency preserving

Decomposition Theorem

Example -

Let R(a,b,c,d) and F = {a→b, a→c, c→d}

Check weather above relation is lossless and dependency preserving if decompose as

a. R1 (a,b,d) and R2 (b,c) :

lossy, no dependency preserving

b. R1 (a,b,c) and R2 (c,d) :

lossless, dependency preserving

END OF UNIT III